数理解析研究所講究録
1084巻 1999年 154-176

渦運動に対する高速数値計算法

名古屋大学大学院多元数理科学研究科　坂上貴之 (Takashi SAKAJO)

1 イントロダクション

無限遠方で0になるような境界条件に対するLaplace方程式

$$\Delta\phi(x) = -f(x), \quad x \in R^d,$$

の解はGreen関数を使って次のような積分で書ける。

$$\phi(x) = \int G(x-y)f(y)dy. \tag{1}$$

今、関数fとして、次のようなN点の点電荷の分布を与える

$$f(y) = \sum_{j=1}^{N} q_j \delta(y - y_j).$$

ただし、q_jは電荷の強さ、y_jは電荷の位置、δはDiracのδ関数である。すると積分(1)は次のような和の形にかける：

$$\phi(x) = \sum_{j=1}^{N} q_j G(x - y_j). \tag{2}$$

ここで与えられたN点$\{x_i\}$におけるポテンシャル$\phi(x_i)$を求めるために必要な計算量は$O(N^2)$となり、Nが大きくなるような大規模数値計算を実際に行なう上での非常な障壁となる。

これに対し、1980年代後半になって(2)を高速に計算するためのアルゴリズム[1, 2, 3]が考案された。これらの方法は一般にTree法と呼ばれ、その計算量を$O(N \log N)$にまで減らすことができる。この方法は関数の近似理論を用いて計算量を減らすので、(2)をある程度の誤差を許して評価する。つまり、これは解析的高速計算法とも呼べるものであり、いわゆるFFTのようなe^{ikx}の代数的な性質を利用して、正確に離散Fourier変換を$O(N \log N)$で計算するような代数的高速算法とは異なる。しかし、逆に近似誤差を解析的に評価することができるため、その誤差を計算機の丸め程度の精度の範囲におさまるようにできれば、実用上計算量だけを減らして計算ができることになる。

その後1987年になって、このTree法の考えを一歩進めてL.GreengardとV.Rokhlinらは(2)の和を$O(N)$で評価する計算方法[4, 5]を提案した。この方法はFast Multipole

種類	代数的高速計算法	解析的高速計算法	
名称	FFT	Tree 法	FMM
アイデア	e^{ikx}の周期性 に注目	粒子とクラスタ間 相互作用の関数近似	クラスタとクラスタ間の 相互作用の関数近似
計算量	$O(N \log N)$	$O(N \log N)$	$O(N)$
近似誤差	なし	あり	あり
適用例	Fourier 変換 convolution の計算	ポテンシャル場の計算 流体運動の計算	ポテンシャル場の計算
近似方法		テイラー展開	多重極展開と局所展開

表 1: $O(N^2)$ の計算を高速に行なう数値計算法

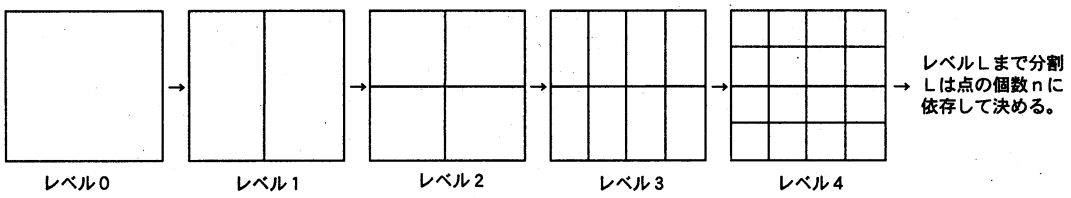
Method(FMM) と呼ばれ、画期的な高速化を達成する計算法として注目されている。このアイデアは熱方程式や Helmholtz 方程式の解の数値計算 [11, 14] にも適用され、大規模な数値計算に対して広く適用できる方法として期待されている。

本報告ではこうした解析的高速計算法の基本的なアイデアおよびそれらの発展の現状 [6, 7, 8, 9, 12]、また実際に応用する上での問題点などについて、第 2 章で述べる。そして、高速計算法の流体運動への適用 [16] について説明を第 3 章以降で行なう。今回扱う問題は 2 次元の周期境界条件を持つ非圧縮・非粘性の渦層の数値計算である。周期境界条件を持つ問題に対して上記高速計算法を単純に適用することはできない。しかし、予めいくつかの処理（前処理）を行なうことで高速計算を利用することができる。この方法について解説し、数値計算結果も併せて紹介する。

2 解析的高速計算法

$O(N^2)$ の計算を高速に行なう算法の一つとして FFT がよく知られている。この方法は $\exp(ikx)$ の周期性（代数的性質）に注目して離散 Fourier 変換を $O(N \log N)$ で行なう方法である。FFT では離散 Fourier 変換の値を誤差なく正確に計算できる。その一方、今回紹介する計算法は高速化を行なう上で、ある種の関数近似を用いるため近似的にしか評価しえない。その意味で前者を代数的高速数値計算法、後者を解析的数値計算法と呼ぶことがある。表 1にその特徴を記している。解析的数値計算法としては Tree 法と FMM の 2 種類があり、それぞれに様々な特徴を持っている。計算量の面からみれば FMM は非常に優れた計算法であるが、適用範囲がせまいなどの欠点もある。このセクションでは Tree 法と FMM の計算原理を簡単に説明し、両計算法の特徴を明らかにする。

(a) 計算領域の分割



(b) 分割した領域に階層構造（2分木構造）を入れる

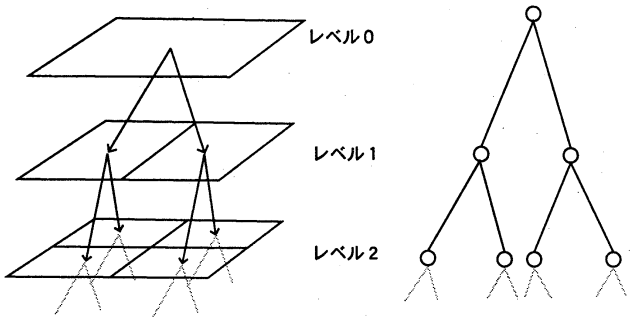


図 1: 計算領域の分割と階層構造

2.1 Tree 法

Tree 法は後述するように計算領域を分割して、そこに2分木構造を入れるためにそのように呼ばれる。詳しいアイデアおよびアルゴリズムについては、Appel[1], Barnes & Hut [2] および Draghicescu[3] を参考にしてほしい。このセクションでは Draghicescu[3] の方法を取り上げて、その計算原理を説明する。説明は簡単のための2次元の場合について行なうが、3次元の場合でも基本的なアイデアは変わらない。

計算領域の分割、2分木構造の導入　図 1(a) を見て欲しい。いま、点電荷が分布している領域を含む矩形領域を考えて、それを以下のようなステップを経て分割を繰り返す。まず、計算領域を縦に2等分する、この縦長方形をレベル1の領域と名付ける。次にこのレベル1のそれぞれの長方形を横に2分割する、その結果4つの矩形領域が生成される。これをレベル2の領域と呼ぶ。以下、同様に各レベルに属する矩形領域を縦横の順に分割をレベル L まで繰り返していく。このレベル L の大きさは与えられた点電荷の数に応じて決める ($L = \log_2 N$)。

さらに、こうして得られた各レベルの領域に階層構造 (2分木構造) を入れる。これにより領域の検索を2分木検索によって高速に (再帰的に) 行なうことが可能になる。(図 1(b))

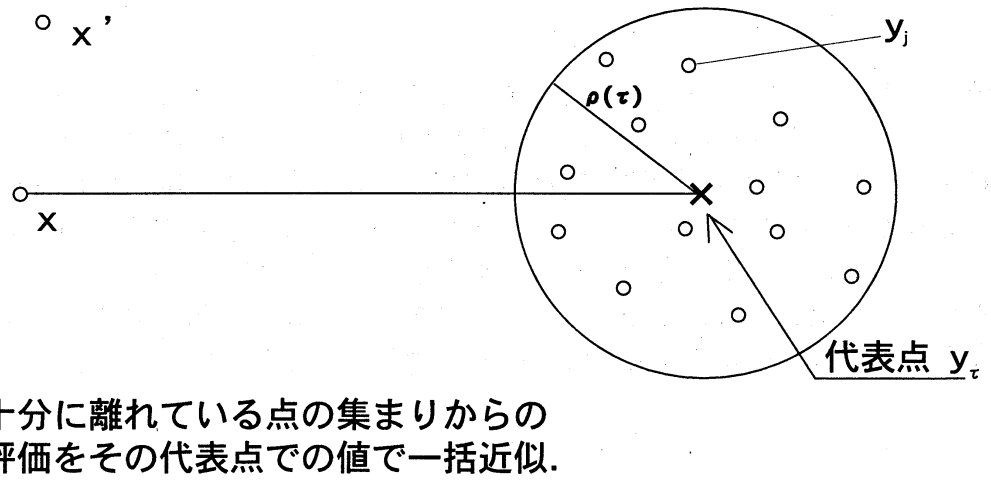


図 2: Tree 法の近似のアイデア：十分離れた点とクラスタの相互作用

粒子クラスタ間の相互作用の関数近似　与えられた点 x と、そこから十分離れたところにある点電荷の集まり (クラスタ) を考える。このクラスタにある点電荷 $\{y_j\}$ からの x に対する寄与を近似する。図 2を参照のこと。クラスタ内に一点だけ代表点をとる。これを y_τ とする。代表点のとり方はクラスタ内の点電荷の重心をとるなど、いくつかの方法があるが、今回紹介する Tree 法では先ほど分割した各矩形領域の対角線の交点をとる。 今、クラスタ内の点電荷からの x におけるポテンシャルへの寄与は十分距離が離れているので、小さいとしてよい。そこで、このクラスタ内にあるすべての点電荷からの寄与をその代表点 y_τからの寄与として一括に近似評価しようというのがアイデアの基本である。

例えば近似にポテンシャル関数の Taylor 展開の有限λ階打ちきりによって近似するとすれば、次のような式が得られる。

$$\phi(x) \approx \sum_{j=1}^{N} \sum_{|k|<\lambda} a_k(x, y_\tau)(y_j - y_\tau)^k = \sum_{|k|<\lambda} a_k(x, y_\tau) \sum_{j=1}^{N} (y_j - y_\tau)^k. \tag{3}$$

a_kは k階の Taylor 係数である。また、この右辺の最後の和

$$\sum_{j=1}^{N} (y_j - y_\tau)^k, \tag{4}$$

は x によらない値であることに注意する。つまり、この和はクラスタに分布する点電荷と、その代表点よってのみ決定されるデータである。以後このデータを “クラスタデータ” と呼

ぶが、あらかじめクラスタごとに、このデータを計算しておけばその再利用が可能になる。すなわち、例えば図2にあるように x の近くに x'なる点があったとする。すると $\{y_j\}$ の集まりであるクラスタはこの x'からも十分遠いと判断できる。よって、近似ポテンシャルを (3) に従って計算することができるが、この時にクラスタによって決まる和 (4) は、既に計算してあるので改めて計算する必要がない。このようにして、近似の計算の手間を大きく減らすことができる。

クラスタデータの計算　Tree 法の高速化の原理をこれまで述べてきたが、実際にどのようなアルゴリズムとしてこれを実装するかが大きな問題である。Tree 法の場合、点のクラスタは最初に定義した分割計算領域の階層構造に含まれる矩形と同一視されるので、どの矩形にどの点電荷が含まれているかを調べて、それに応じて和 (4) を計算しなければならない。これを行なう際に 2 分木検索が大きな役割を果たす。

まず、電荷を一つ選び（これを y_jとする）それをレベル 0 の矩形に含まれるかチェックする。レベル 0 は全計算領域なのでこの点電荷を含む。したがって、次の量

$$(y_j - y_0)^k, (k = 1, ..., \lambda).$$

を計算する。ただし、y_0はレベル 0 の矩形の中心である。次に、2 分木検索にしたがって、y_jを含むレベル 1 の矩形を選ぶ。それに対しても、先ほどのように $(y_j - y_\tau)^k$を計算する。このステップを 2 分木検索の要領で再帰的にレベル L まで行なえば、y_jを含むすべての矩形に対してクラスタデータが計算される。これを他の N点に対しても同様に行なえば、結局すべての点に対して和 (4) が計算できることになる。この計算は階層 $L = \log_2 N$の 2 分木探索を N点に対して行ない、各段階でλ階のクラスタデータを計算するので計算量は $O(N\lambda^2 \log_2 N)$ となることに注意する。

遠近クラスタの決定　次に行なうべきは、与えられた点に対して近似によって評価する遠いクラスタ (Far Field) と近いために直接に評価する近いクラスタ (Near Field) を探すことである。この時も 2 分木探索によって、これらの遠近クラスタを選び出す。具体的には次のような手順で行なう。まず、クラスタと点の「遠近」を判断する条件を定義しなければならない。これは近似の精度や計算量に大きく関係するので、注意深く選ぶ必要がある。これを以後「遠近条件」と呼ぶ。具体的な条件については Draghicescu[3] を参照のこと。図 3を見ながら説明を行なう。始めに、与えられた点とレベル 0 の矩形の中心に対して、この遠近条件を確認する。この場合は矩形の中に与えられた点自身が含まれているので遠いクラスタとはなりえない。そこで、条件を満たさなかった矩形に対して、自分の子供矩形 (レベル 1) に対して再び遠近条件をチェックする。図 3では、双方とも条件を満たさない。次にその子供 (レベル 2) の各矩形に対しても条件をチェックする。ここで灰色に塗りつぶされた矩形が条件を満たしたとする。この条件を満たした矩形の中にある点電荷からの寄与は、この矩形の代表点からの関数近似によって評価するため、この矩形の子供に対してはもう遠近条件をチェックする必要はない。 この次には遠近条件を満たさなかった残された 3 つの矩形のレベル 3 にある子供に対して遠近条件を確認する。このようにして、遠近

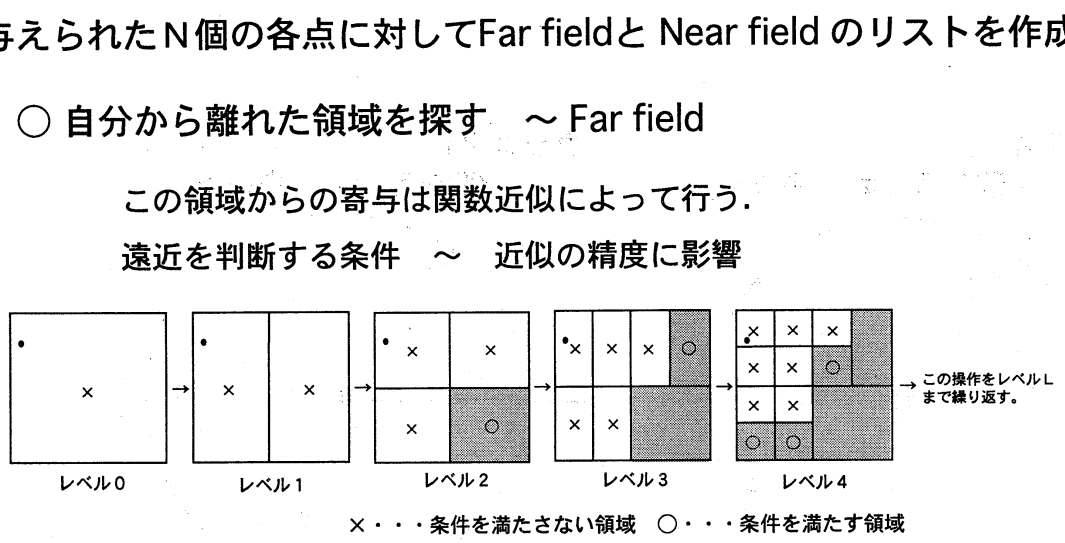


図 3: 遠近クラスタの選定

条件を満たす矩形をレベル L まで繰り返すと結局与えられた点に対する遠いクラスタ (Far Field と呼ぶ) とそうでないクラスタ (Near Field と呼ぶ) が残されることになる。この Far Field に関しては近似式 (3) によって近似評価し、Near Field に含まれる点電荷に関しては直接ポテンシャルを計算する。Draghicescu[3] によれば、Far Field の数と Near Field に含まれる点の数が評価されており、このステップで必要な計算量は $O(N\lambda^2 \log_2 N)$ であることが示されている。

Tree 法の近似誤差と計算量の見積り　Draghicescu[3] によって、この算法の近似誤差と計算量が与えられている。数値計算に現れるパラメータは Taylor 展開の近似階数、遠近条件および与えられた点電荷の数であるが、近似階数や遠近条件をうまく選ぶと近似誤差は $O(\frac{1}{N})$、計算量は $O(N(\log_2 N)^3)$ になることが示されている。

2.2 Fast Multipole Method

Tree 法は点とクラスタの相互作用についての考察が基本になっている。したがって、この算法の適用範囲は距離に反比例して値が減少する無限階微分可能な関数であればよい。一方、もともとの問題が Laplace 問題で与えられており、その結果として取り扱う関数が調和関数であるということを考慮すれば、クラスタとクラスタの間の相互作用をうまく利用して、$O(N^2)$ の計算を $O(N)$ にまで減らすことができるという解析的高速計算法が Greengard

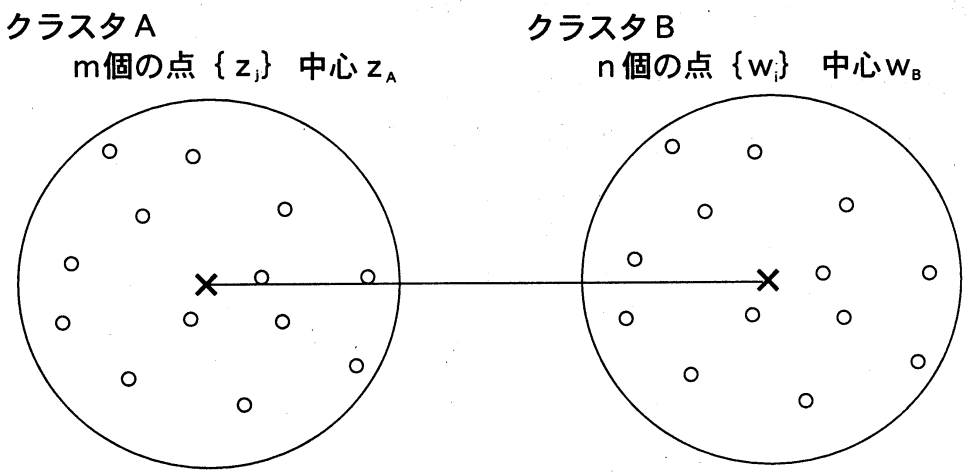


図 4: FMM の問題設定

と Rokhlin[4] によって与えられた。この算法は Fast Multipole Method(FMM) と呼ばれ新しい高速解法として注目されている。このセクションでは FMM の基本的なアイデアとその実装について述べる。以下では 2 次元の FMM[4] について述べるが、3 次元の場合 [5] も基本的なアイデアは同じである。

問題の設定　以後の説明のために問題の設定を次のように行なう。FMM ではクラスタとクラスタの相互作用を考えるので、今二つの点電荷の集まりを用意し、それぞれクラスタ A、クラスタ B と呼ぶことにする。クラスタ A には m 個の点電荷が位置 $z_i, (i = 1, ..., m)$(大きさ q_i) に分布しており、その中心の位置を z_Aとする。一方、クラスタ B の方には n 個の点電荷が位置 $w_j, (j = 1, ..., n)$ に分布し、その中心を w_Bとする。そうしておいて、クラスタ B の各点電荷におけるクラスタ A にある点電荷からの寄与を考える。

多重極展開　2 次元の点電荷の場合、2 次元空間と複素平面を同一視して、位置 z_0にある点電荷 (大きさ q) の作るポテンシャル$\phi(z)$ は次のように与えられる。

$$\begin{aligned}\phi(z) &= q\log(z - z_0) \\ &= q\log z - q\sum_{k=1}^{\infty}\frac{1}{k}(\frac{z_0}{z})^k, (|z| > |z_0|).\end{aligned}$$

そこで、まずクラスタ A の中心 z_A に関して次のような展開を与える。

$$\begin{aligned}\phi_{z_A}(z) &= a_0 \log(z - z_A) + \sum_{k=1}^{\infty} \frac{a_k}{(z - z_A)^k}, \\ a_0 &= \sum_{j=1}^{m} q_j, \\ a_k &= \sum_{j=1}^{m} \frac{-q_j (z_j - z_A)^k}{k}.\end{aligned}$$

この展開をクラスタ A に関する多重極展開とよぶ。実際の計算ではこの第 2 項の無限和を p 階で打ち切ってポテンシャルの近似値とする。この時、多重極展開の係数は a_p まで求めるのでこの操作のために必要な計算量は $O(mp)$ である。

多重極展開から局所展開への変換　上記操作によって求めた、クラスタ A に関する多重極展開 ϕ_{z_A} を次にクラスタ B に関する局所展開 Ψ_{w_B} に変換する：

$$\phi_{z_A}(z) = a_0 \log(z - z_A) + \sum_{k=1}^{p} \frac{a_k}{(z - z_A)^k} \quad \rightarrow \quad \Psi_{w_B}(z) = \sum_{k=1}^{p} b_k (z - w_B)^k.$$

実際の計算では係数 a_k から b_k への変換公式が与えられることになることに注意。

そうしておいて、クラスタ B にある点でのクラスタ A からのポテンシャルの寄与はこの局所展開の係数を用いて $\Psi_{w_B}(w_j), (j = 1, ..., N)$ で計算する。この計算量は $O(np)$ である。これと同じ操作をクラスタ B によるクラスタ A への寄与を求める時にも行なうことができて、結局それぞれの相互作用を $O(mp + np)$ で求めることができる。これが FMM の基本的な計算原理である。

4 つの変換公式　FMM のアルゴリズムを構成する時には以下のような 4 つの変換公式が重要になる。

1. 多重極展開（中心 z_0）

$$\phi_{z_0} = a_0 \log(z - z_0) + \sum_{k=1}^{p} \frac{a_k}{(z - z_0)^k}, \; a_0 = \sum_{k=1}^{m} q_j, \; a_k = \sum_{j=1}^{m} -\frac{q_j (z_j - z_0)^k}{k}.$$

2. 多重極展開の中心シフト演算（中心 z_0 から原点へ）

$$\phi_{z_0} = a_0 \log(z - z_0) + \sum_{j=1}^{p} \frac{a_k}{(z - z_0)^k} \quad \rightarrow \quad \phi_0 = a_0 \log z + \sum_{l=1}^{p} \frac{b_l}{z^l},$$

$$b_l = \sum_{k=1}^{l} a_k z_0^{l-k} \begin{pmatrix} l-1 \\ k-1 \end{pmatrix} - \frac{a_0 z_0^l}{l}. \qquad \begin{pmatrix} l \\ k \end{pmatrix} \text{は 2 項係数。}$$

3. 多重極展開から局所展開への変換 （中心 z_0から原点へ）

$$\phi_{z_0} = a_0 \log(z - z_0) + \sum_{k=1}^{p} \frac{a_k}{(z - z_0)^k} \quad \rightarrow \quad \Psi_0 = \sum_{l=0}^{p} b_l z^l,$$

$$b_0 = \sum_{k=1}^{p} \frac{a_k}{z_0^k}(-1)^k + a_0 \log(-z_0), \qquad b_l = (\frac{1}{z_0^l} \sum_{k=1}^{p} \frac{a_k}{z_0} \begin{pmatrix} l+k-1 \\ k-1 \end{pmatrix} (-1)^k) - \frac{a_0}{l z_0^l} (l \geq 1).$$

4. 局所展開の中心のシフト演算 （中心 z_0から原点へ）

$$\Psi_{z_0} = \sum_{k=1}^{p} a_k (z - z_0)^k \quad \rightarrow \quad \Psi_0 = \sum_{l=0}^{p} (\sum_{k=l}^{p} a_k \frac{k}{l} (-z_0)^{l-k}) z^l.$$

2番と4番は、それぞれ多重極展開と局所展開の中心を任意の中心に移すための公式である。これはいままで説明してきたFMM近似原理の中には登場しなかったが、実際にアルゴリズムを構成する際には重要な役割を果たす。もちろんのことであるが、各変換の可能性や和の収束半径および p 階打ちきりによる近似誤差などを見積もる必要があるが、詳しい誤差評価などは原論文 [4] などを参考にしてもらうことにして、ここでは述べない。FMMについては上記4つの公式が近似の誤差も含めて、しっかりと構成できるような関数であれば、以下に説明するようなアルゴリズムによって $O(N)$ の計算を達成できることに注意。

計算領域の分割と多重極展開係数の計算　図5を見て欲しい。FMMでもTree法と同様に計算領域を設定して、それを分割して階層構造をいれる。各レベルでの各分割領域のことを以下では「セル」と呼ぶことにする。ただし、Tree法では2分木探索を行なうために縦と横を別々に分割したが、FMMの場合は縦横同時にそれぞれ2等分して次のレベルとする点が異なっている。まずこの各レベルの各分割セルに対して、多重極展開係数を求める。この時に多重極展開のシフト演算を用いる。つまり、まず最下層レベルにあるセル（これは N個になるようにとってある。）の中心での多重極展開を求める。それから一つ上のレベルのセルでの多重極展開を求めるのに、そのセルの4つの子供セル（今の場合は最下層レベルのセル）での多重極展開係数を自分の中心の展開にシフトして足し合わせるという操作を行なう。これを再帰的に上のレベルに繰り返していくことによって、すべてのレベルのすべてのセルでの多重極展開が求められる。このステップをUpward Passと呼ぶ。

多重極展開から局所展開への変換プロセス　次に求めるのは最下層レベルでの各セルの中心についての局所展開である。その局所展開係数さえ計算できれば、それぞれの点でのポテンシャルが簡単に計算できる。 まず任意のレベル（ただし、レベル2以上）の任意のセルに対して次のような領域が定義できる。

- 自分に隣接しているセル。
- 自分とは隣接していないが、互いの親領域どうしは隣接している。
- 自分とも隣接していないし、かつその親同士も隣接していない。

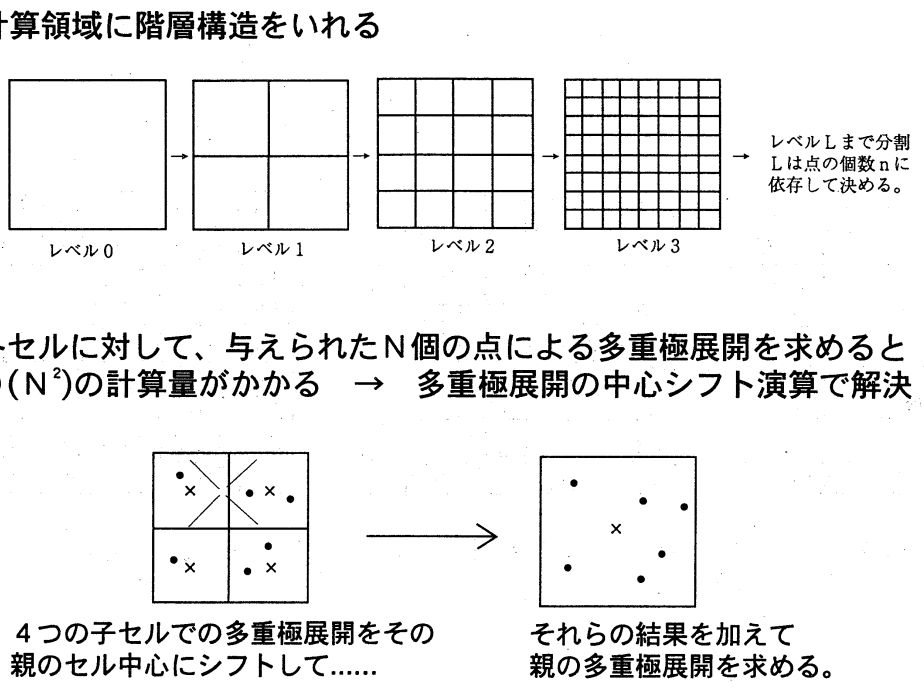


図 5: 計算領域の分割と多重極展開の計算 (Upward Pass)

それぞれの領域に対して、以下のような評価を行なう。

- 評価しない。(近いセルなので多重極近似はできない。)
- このセルでの多重極展開を局所展開に変換して加える。(多重極→局所変換)
- 自分の親の局所展開を自分の中心についての局所展開にシフトする。(局所→局所変換)

図 6を見ながら説明しよう。レベル 2 からレベル 4 までの分割に対して上のステップをを当てはめてみる。まずレベル 3 に対して黒く塗りつぶされた領域での局所展開を考える。まず自分と接している 8 つのセルは白抜きになっている。次に自分とは接していないが親同士が接しているセルは灰色に塗りつぶしている。さらに、自分とも親どうしも接していないセルについては波模様で塗りつぶしている。この 3 つの領域に対して、まず灰色の部分は自分からは遠いセルということで、そこでの多重極展開を自分を中心とする局所展開に変換する。次に波模様のセルに対しては、自分のレベル 2 での親セルからの局所展開を自分の中心についての局所展開に変換する。この操作によって、なぜ波模様のセル群からの寄与を計算できるのかと言えば、それらのセルは実はレベル 2 において、自分の親セルの灰色領域と判断され、その時にすでに局所展開として自分の親の方にその効果が織り込まれているからである。あとは親の局所展開をシフトして自分の中心にうつしてやればよい。このステップを順に下のレベルまで繰り返すことによって、最下層まで行けば最終的

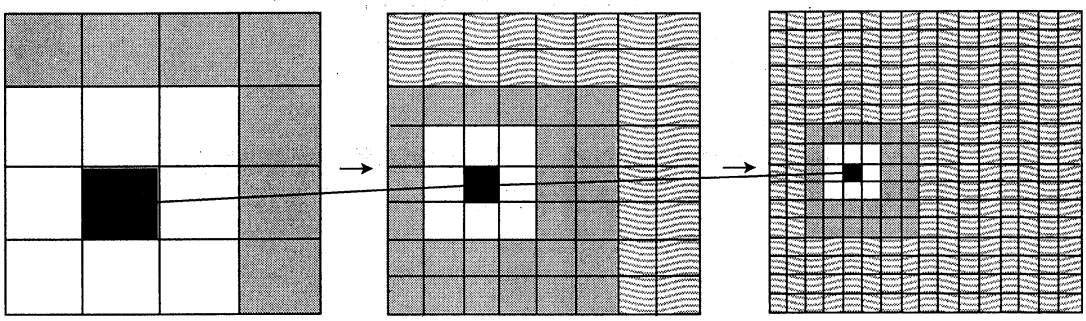


図 6: 局所展開の計算 (Downward Pass)

に最下層レベルでの各セルにおける局所展開係数が求められる。このステップは先ほどの Upward Pass に対して Downward Pass と呼ばれている。

後は各点電荷について、自分を含む最下層レベルのセルでの局所展開を使って十分離れたクラスタからの寄与を計算し、自分の隣接セルにある点電荷からの寄与は直接計算すればよい。これによって、すべての点からの寄与を計算することができる。

誤差評価と計算量 Greengard と Rokhlin[4] によれば、このアルゴリズムで計算した場合、計算量 $O(N)$、近似誤差を $(\frac{1}{2})^p$で計算できることを示している。つまり、p が比較的小さくても、その近似誤差は非常に小さいので精度が容易に改善される。逆にいえばその精度を計算機の丸めまでとれば、計算機で直接和をとって計算するのと何の誤差もなく評価できることがわかる。

2.3 FMM の研究の現状

Greengard & Rokhlin [4] 以降、この算法自身は実に多様な発展を遂げている。3 次元の FMM[5] は関数$\frac{1}{r}$の球面調和関数展開をもとにして達成されている。また熱方程式の解に現れる Gauss 変換に対しても、Hermite 多項式系の母関数表示を用いて FMM[11, 13] が構成されている。また、このアルゴリズムの高速改良も進んでいる。まず 2 分木探索などを用いず自分の親子のセル関係を巧みに用いるために、並列計算機への応用 [7] が容易である。

つぎに、点電荷の分布に応じて高速化効率を挙げるadaptive法[6]などもある。

この計算法は算法としては画期的なものであるが、問題もある。まず、上記で示したような4つの変換公式がどのようなポテンシャルに対しても構成できるかどうかがわからない。たとえそのような変換公式が構成できたとしても、多重極展開をすること自体に時間がかかるようであれば、それだけで全体の効果は薄れてしまう。実際に3次元FMMではそのこと自体が非常にボトルネックになっている。この3次元の多重極展開とその局所展開に対する高速技法[8, 9]が、いくつか考案されているが、本質的にFMMの困難が取り去られたわけではない。また、今回後半部で説明する渦運動の数値計算においても、扱う関数が調和関数ではないためにFMMの使用は行なわなかった。FMMが非常に優れた方法でありながら、応用の上でその能力に見あっただけの利用が報告されていないのは、こうしたアルゴリズムの性質および実装の困難にもよることに注意して欲しい。

なお、3次元FMMの実用的な高速化を目指して、1997年にActa Numericaに掲載されたGreengardとRokhlinの論文[10]はこれまでのFMMの発展状況や新しい実用か手段について詳しく書かれているので、FMMを実用的な計算を試みる場合に一読する価値がある。

3 渦運動と高速計算法

著者が研究で用いた具体的な流体運動への高速解法への応用とそのための前処理について説明する前に、このセクションでは流体力学においてFMMやTree法が必要になる理由を簡単に説明する。2次元の非圧縮・非粘性Euler方程式の 渦度(ω) −流れ関数(Ψ)表示は次のように与えられる。

$$\begin{aligned} \partial_t \omega + (u \cdot \nabla)\omega &= 0, \\ \Delta\Psi = -\omega, \qquad & u = (\Psi_y, -\Psi_x). \end{aligned}$$

ここでΨに関するLaplace方程式を解くと次を得る。

$$\begin{aligned} \Psi(x) &= \int G(x-y)\omega(y)dy, \\ G(x) &= -\frac{1}{2\pi}\log|x|. \end{aligned}$$

よって、速度場は以下のように与えられる。

$$\begin{aligned} u(x) &= \int K(x-y)\omega(y)dy, \\ K(x) &= \frac{1}{2\pi}\frac{(-x_2, x_1)}{|x|^2}. \end{aligned}$$

これだけ準備をしておいて、次に2次元空間内に渦度がある有界領域Ωの上に分布していると仮定する(図7上)。その領域内にある点をaとしてこのaの運動を考える。その点の位置を$z(\alpha, t)$とする。αは粒子とともに動く座標系でのLagrangeパラメータ、tは時間

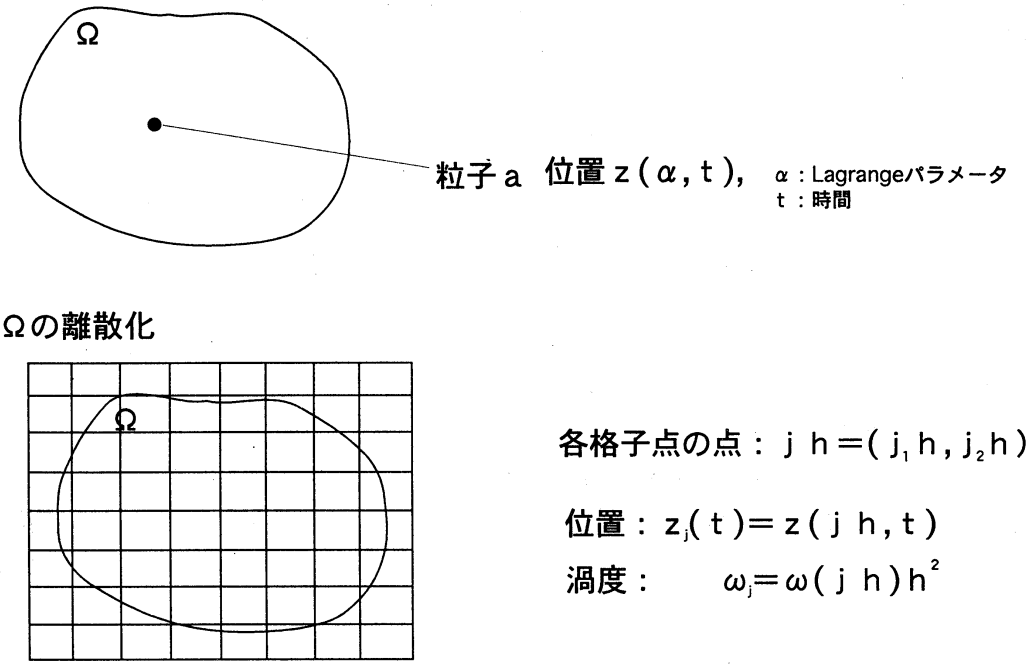


図 7: 渦の領域とその離散化

である。2 次元の流体運動の場合、渦度は粒子の起動に沿って保存されるから、この粒子の運動方程式は

$$\begin{aligned} \frac{dz}{dt} &= \int K(z - z(\alpha', t))\omega(z(\alpha', t), t)d\alpha' \\ &= \int K(z - z(\alpha', t))\omega(z(\alpha', 0), 0)d\alpha' \end{aligned} \tag{5}$$

で与えられる。この方程式を離散化して流体（渦領域）の数値計算を行なう。

離散化の方法としてまずΩを幅 h の格子点に分割する (図 7下)。各格子点の代表点を $jh = (j_1h, j_2h)$ としてその格子点での渦度の大きさを$\omega_j = \omega(jh)h^2$とする。この格子点の代表点を流体とともに移動する Lagrange 粒子と見てその位置を $z_j(t) = z(jh, t)$ とすると積分 (5) は次のような離散常微分方程式系になる。

$$\frac{dz_i}{dt} = \sum_j K(z - z_j)\omega_j,\ z_i(0) = ih.$$

各点に対する、この速度場がちょうどイントロダクションで説明した 2 重和の計算量 $O(N^2)$ を要求する。このような数値計算方法を Point Vortex Method と呼んでいるが、この速度場を計算する段階ではまだ調和関数を扱っているので、FMM や Tree 法の双方が利用可能である。

4 周期境界条件を持つ渦層の高速計算

4.1 渦層とその支配方程式

このセクションでは、実際の流体運動に解析的高速計算法を適用することを考える。具体的に我々が取り扱う問題は2次元の渦層の運動である。渦層とは流体速度の不連続面として定義され、その不連続面の上にだけ渦度が分布しているような理想的な物理モデルである。この面以外の領域で流体はポテンシャル流である。2次元の不連続面は曲線なので、2次元空間と複素平面を同一視して次のように定義できる。

$$z(t,\Gamma) = x(t,\Gamma) + \mathrm{i}y(t,\Gamma).$$

ただし、t は時間、Γは曲線に沿った流れとともに移動する Lagrange パラメータである。この時、渦層の運動は Birkhoff-Rott 方程式と呼ばれる方程式で記述される：

$$\frac{\partial z^*}{\partial t} = \frac{1}{2\pi\mathrm{i}}\int_{-\infty}^{\infty}\frac{d\Gamma'}{z(t,\Gamma)-z(t,\Gamma')}.$$

$*$は複素共役を表し、左辺の積分は Cauchy の主値積分である。今回の問題では渦層に周期境界条件を課す。つまり条件

$$z(t,\Gamma+1) = z(t,\Gamma)+1,$$

を考える。この条件のもとで Birkhoff-Rott 方程式は以下のように書き換えられる。

$$\begin{aligned}\frac{\partial z^*}{\partial t} &= \int_0^1 K(z(t,\Gamma)-z(t,\Gamma'))d\Gamma',\\ K(x+iy) &= -\frac{1}{2}\frac{\sinh(2\pi y)+\mathrm{i}\sin(2\pi x)}{\cosh(2\pi y)-\cos(2\pi x)}.\end{aligned} \tag{6}$$

以後、この方程式を数値計算する。

4.2 渦法とその高速数値計算

Birkhoff-Rott 方程式は Ill-posed であることが Caflisch[17] によって示されており、また滑らかな初期値に対する解に有限時間で特異点が発生する [19] ということも知られている。そのため方程式を単純に離散化しただけでは数値計算はうまくいかない。ここで渦法と呼ばれる数値計算法を用いる。つまり (6) の関数 Kの変わりに、ある正の十分小さなδを導入して非特異化 Birkhoff-Rott 方程式（以下δ方程式と呼ぶ）を扱う；

$$\begin{aligned}\frac{\partial z^*}{\partial t} &= \int_0^1 K_\delta(z(t,\Gamma)-z(t,\Gamma'))d\Gamma',\\ K_\delta(x+iy) &= -\frac{1}{2}\frac{\sinh(2\pi y)+\mathrm{i}\sin(2\pi x)}{\cosh(2\pi y)-\cos(2\pi x)+\delta^2}.\end{aligned}$$

このδ方程式は well-posed である。$\delta \to 0$ の時の解の収束性は特異点が生成するまでは強い意味での収束 [18] が、特異点が生成してから後については、L^2での弱収束列を構成すること [20] がわかっている。

この右辺の積分を台形公式で離散化すると、次の常微分方程式系を得る。

$$\frac{dz_n^*}{dt} = \frac{1}{N}\sum_{m=1}^{N} K_\delta(z_n(t) - z_m(t)), (n = 1, 2, ..., N). \tag{7}$$

この和を計算するのに $O(N^2)$ の計算量が要求される。ここで第 2 章で述べた解析的高速数値計算法を使用する。しかし、この (7) の積分核 K_δをよく見ると、これはδがあるために調和関数ではない。よって多重極展開を構成することができない。このことから我々が選択できるのは Tree 法だけということになる。しかしながら実は Tree 法を応用する上でも非常な困難が生じる。まず K_δ関数が周期関数を含んでおり、遠方で値が小さくならないので単純な Tree 法の応用ができない。さらにこの関数の高階 Taylor 展開という操作自身が非常に繁雑になり、それだけに非常に時間を費やす。その結果、高速計算の利得が得られないことになる。

このような困難を取り除くために、我々は方程式を次のような意味で「前処理」した。すなわち等角写像によって変数変換する；

$$w = \exp(2\pi i z).$$

その結果、方程式は次のようになる。

$$\begin{aligned} \frac{\partial w^*}{\partial t} &= \int_0^1 K'_\delta(w(t,\Gamma), w(t,\Gamma')), \\ K'_\delta(w,v) &= -\pi i w \frac{(w+v)(w^*-v^*)}{|w-v|^2+\delta^2}. \end{aligned}$$

この方程式を再び台形公式によって離散化する。

$$\frac{dw_n^*}{dt} = \frac{1}{N}\sum_{m=1}^{N} K'_\delta(w_n, w_m).$$

この右辺の和の積分核 K'_δを見ると、これに Tree 法を適用する場合、結局$\frac{1}{r}$の Taylor 展開ができればよいことがわかる。この展開は簡単な再帰公式が与えられているので、高速に Taylor 係数を計算することができて、実用上の高速算法の適用が可能である。

今回の計算の初期値はフラットな 2 次元渦層（Birkhoff-Rott 方程式の定常解）に僅かな摂動を加えたものである。実際にはこれを等角写像で w平面に移したものを用いる。すなわち

$$\begin{aligned} z(0,\Gamma) &= \Gamma + \frac{1}{2}\mathrm{i} - 2\pi\epsilon(1-\mathrm{i})\sin(2\pi\Gamma), \\ w(0,\Gamma) &= \exp(2\pi \mathrm{i} z(t,\Gamma)). \end{aligned}$$

時間方向離散化は 4 次の Runge-Kutta 法を用いた。

λ	時間	効率	相対誤差
4	3	17.7	4.62e-02
6	4	13.3	3.10e-03
8	5	10.6	3.62e-04
10	6	8,83	3.15e-05
12	8	6.63	4.11e-06
14	10	5.30	4.55e-07
∞	55	—	———

表 2: Tree 法の Taylor 展開の近似階数λと相対誤差、効率。$N = 4096, \delta = 0.03$ に固定。

4.3 高速化の効果

ここでは Tree 法の高速化の効果を示す。そのためにまず、いくつかの定義をしておくまず相対誤差を以下のように与える。

$$(\text{相対誤差}) = \max_{1 \le n \le N} \frac{|u_n^f - u_n^d|}{|u_n^d|}$$

ただし、u_n^fは高速計算法で計算した時の速度場を、u_n^dは直接計算によって計算した時の速度場を表す。次に高速化の “効率” を

$$(\text{効率}) = \frac{\text{直接法の計算にかかる時間}}{\text{高速法の計算にかかる時間}}$$

と定義する。これにより（効率）が 1 より大きければ、高速計算法の効果があると判断する。ただし、時間は速度場を一度計算するのに必要な時間を計測する (単位は秒)。

数値計算において用いられるパラメータは以下の 3 つである。

- N:渦法における離散化渦点の数
- δ:渦法の非特異化パラメータ
- λ:Tree 法の Taylor 展開の近似階数

まず離散化渦点の数 Nとδを固定して、Taylor 展開の近似階数を変えた時の効率と相対誤差の変化を表 2に示す。12 階近似で 6.63 倍の高速化が達成され、相対誤差は $4.11e-06$ となっている。表からわかるように近似の階数が大きくなると誤差は小さくなってゆく、一方で効率は少しずつ悪くなってゆく。次に近似階数とδを固定して離散化渦点の数 Nを変えた時の効率と誤差の様子を表 3に示す。渦点の数が増えると効率が劇的に改善される。65536 点で約 3 0 倍の高速化に成功している。さらに相対誤差も Nの増加につれて改善されている。最後に今回行なった変数変換の前処理の効果について表 4に示す。λは 8 階まで近似した。変数変換を行なう前のオリジナルの Birkhoff-Rott 方程式に Tree 法を適用した場合、

N	時間 (直接)	時間 (高速)	効率	相対誤差
4096	53	10	5.30	4.54e-07
16384	886	72	12.3	1.01e-07
65536	14200	472	30.1	1.19e-08

表 3: 離散化渦点の個数 N と相対誤差、効率。$\lambda = 14, \delta = 0.03$ に固定。

N	効率 (前処理有)	相対誤差		効率 (前処理無)	相対誤差
4096	10.6	1.01e-07		0.89	1.26e-05
16384	23.3	1.19e-08		2.29	6.63e-06

表 4: 前処理の効果 (右：前処理あり、左：前処理なし)。ともに近似階数λは 8 。

4096 点では効率が 1 を下回っており、直接法で計算した方が速いことがわかる。16384 点になって、ようやく効率が 1 を超えるものの、前処理を施した方に比べて、その効果は小さいことがわかる。このような差がでるのは、Taylor 展開の係数計算にかかる時間が処理前と処理後では異なることと、処理前の方程式に Tree 法を適用する時に周期境界条件を考慮してアルゴリズムを改良する必要があり、その結果として高速化のメリットが失われることによる。

4.4 渦点の初期分布に関する前処理

渦法とその高速数値計算法によって、時間発展を時刻 0.0 から 10.0 まで計算した結果が図 8(a) である。表示は 2 周期分書いてある。計算パラメータは $N = 4096, \delta = 0.03, \lambda = 12$ にとってある。始めほとんど平らだった渦層が時間が経過すると渦巻状に巻き上がり、それが発達していくが、ここで時刻 8.0 での解の様子を見ると端のあたりで解の精度が落ちていることがわかる。この精度の悪化のため時刻 10.0 での解は信頼できないものとなっている。そこで、4096 点の渦点が渦層の中でどのように分布しているかを知るために、隣合う離散渦点どうしの距離を図 8(b) に示した。この結果からわかるように時刻 8.0 で端の渦点間の距離が大きく伸びていることがわかる。つまり、最初は等分配置した渦点も時間発展するにつれ端の方で距離が伸び、螺旋の真中に点が集まっていくのである。

そこで、離散化渦点の個数を最大限に利用するために、初期の渦点の分布を、始めから端の方にたくさんの渦点が集まるようにしておけばよい。そのために今度は渦層に沿った変数Γの変数変換を考える。変数変換の関数を以下のように与える。

$$\begin{aligned} \Gamma &= \Psi(x), \\ \Psi(x) = \frac{\int_0^x \phi(t)dt}{\int_0^1 \phi(t)dt} &\quad , \quad \phi(t) = \frac{1}{1 - a^2(t - 0.5)^2}, \end{aligned}$$

$$\Psi \in C^{\infty}[0,1], \Psi(1) = 1 \quad , \quad \Psi(0) = 0, \Psi'(0) = \Psi'(1) = 0.$$

たたし、a は分布を制御するパラメータである。今回は $a = 25$ を用いた。その時の$\Psi(x)$ の様子を図 9に示す。この時、積分は

$$\int_0^1 K_\delta(w(\Psi(x),t), w(\Psi(x'),t))\Psi'(x)dx',$$

となり、それを離散化すると以下のような重みつきの和を得る。

$$\frac{1}{N}\sum_{m=1}^{N} K_\delta(w_n, w_m)\Psi'(w_m),\ w_n = w(\Psi(x),t).$$

このような処理を施して得た同条件での計算結果が図 8(c) に示してある。これをみると時刻 8.0 において、端での解の精度の悪化はなくなっている。図 8(d) を見れば、端での渦点の距離も小さいままである。なお、時刻 10.0 においてはやはり端での精度が悪化し始めているが、この精度の悪化は渦点の数を増やすことによって対処した。

4.5 計算結果

高速計算法を渦運動に適用する際の前処理について述べるのが目的であるので、ここでは時刻 0 から 11.5 までの数値計算した結果のみを示す。高速数値計算を実際に巧みに応用することによって、これまでスーパーコンピュータを駆使しても到達が困難だった時刻までの数値計算がワークステーションレベルで達成できるようになったことだけは強調しておく。渦点の数は 65536 点用いた。流体運動としての新しい発見については論文 [16] を参照のこと。

5 まとめ

$O(N^2)$ の計算量を要求する数値計算に対して、それを近似誤差を許しながら高速に評価する解析的高速算法である Tree 法と FMM についてのサーベイを行なった。同時に両計算法それぞれについて適用の範囲や近年の発展の様子なども合わせて述べた。

流体運動において、この解析的高速数値計算法が必要な理由を簡単に述べ、実際に周期境界条件をもつ 2 次元渦層の数値計算に Tree 法を適用することで非常に有意義な結果がでることを報告した。この際に 2 度の変数変換を前処理を施すことで、計算の効率をさらに上昇させることに成功した。

謝辞

今回の招待講演に際し、名古屋大学大学院工学研究科計算理工学専攻 杉原正顯教授にはたいへんお世話になった。ここにその感謝の意を表する。

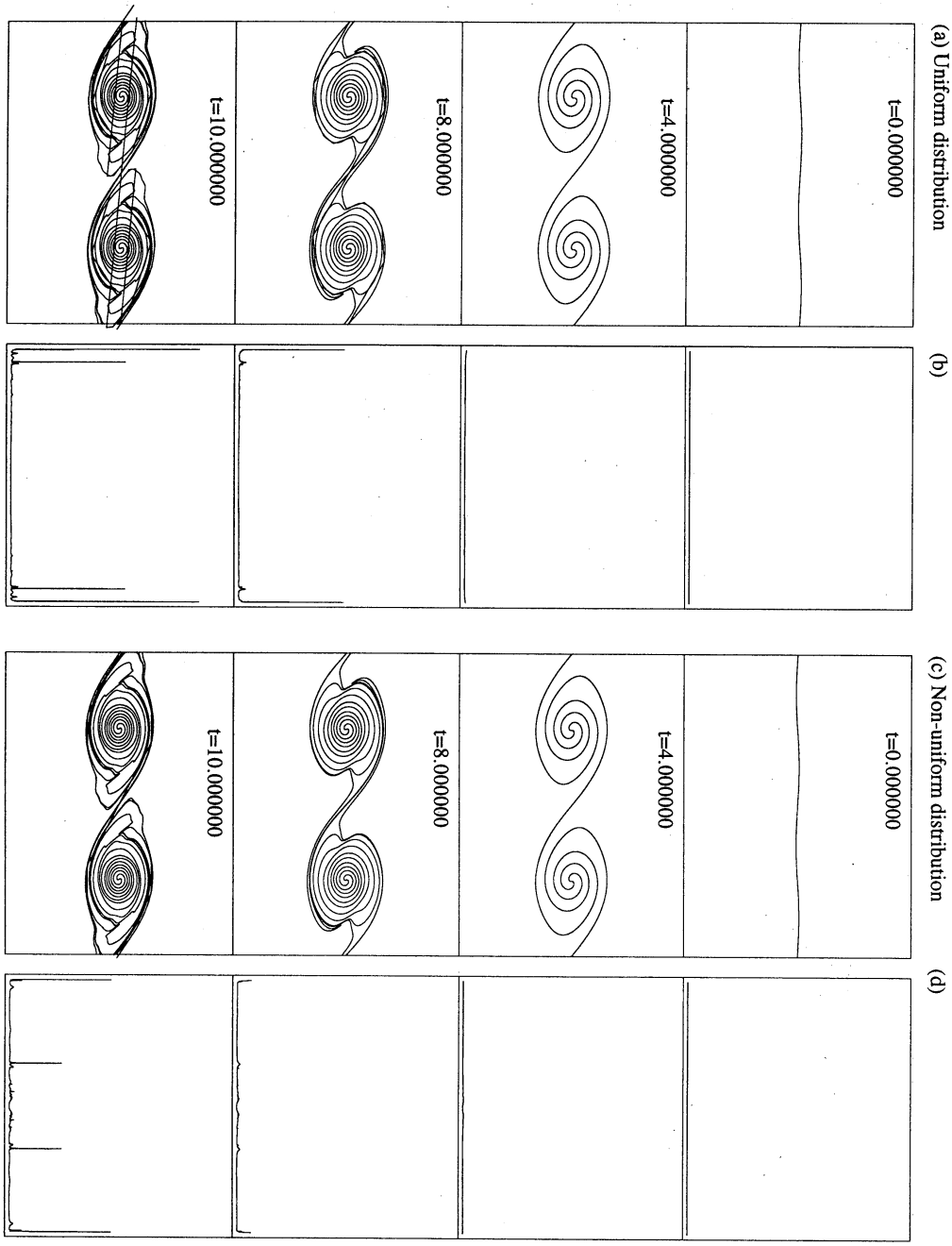


図 8: 初期渦点の分布の変更の成果。(a) 変更前の数値解、(b) 変更前の隣合う渦点の距離。(c) 変更後の数値解、(d) 変更後の隣合う渦点の距離

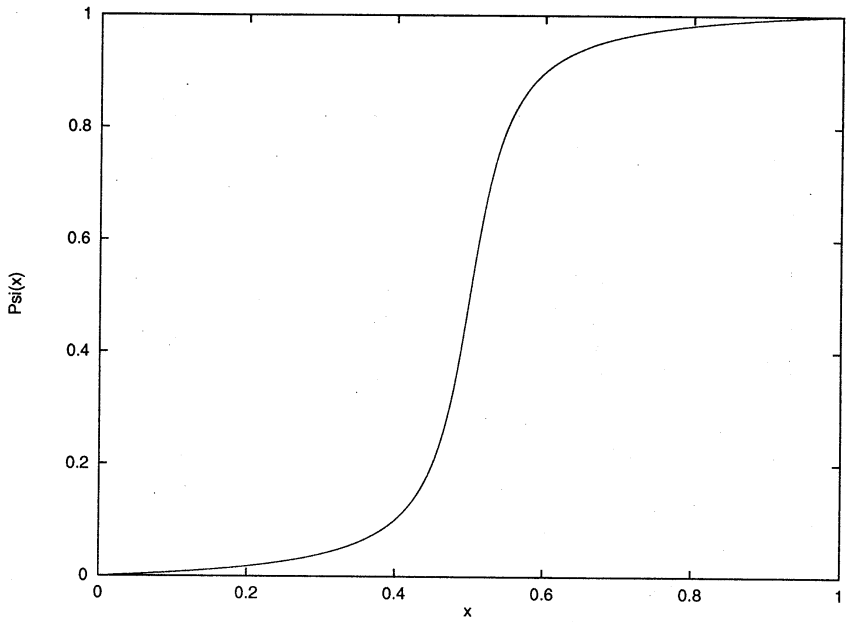


図 9: 渦点の分布を変える変換関数

参考文献

[1] A.W.Appel, An efficient program for many-body simulation, SIAM J. Sci. Stat. Comput., vol.6, pp.85–103 (1985).

[2] J.Barnes and R.Hut, A hierarchical $O(N \log N)$ force-calculation algorithm, Nature, vol.324, pp.446–449 (1986).

[3] C.I.Draghicescu, An efficient implementation of particle methods for the incompressible Euler equations, SIAM J. Numer. Anal., vol.31, No.4, pp.1090–1108 (1994).

[4] L.Greengard and V.Rokhlin, A fast algorithm for particle simulations, J.Comp.Phys., vol.73, pp.325–348 (1987).

[5] L.Greengard and V.Rokhlin, The rapid evaluation of potential fields in three dimensions, Springer Lecture Note #1360, Vortex Methods, eds. C.Anderson and C.Greengard, pp.121-141 (1988).

[6] J.Carrier, L.Greengard and V.Rokhlin, A fast adaptive multipole algorithm for particle simulations, SIAM J. Sci. Stat. Comput., vol.9 no.4, pp.669-686 (1988).

[7] L.Greengard and W.Gropp, A parallel version of the fast multipole method, Computers Math. Application, vol.20 no.7, pp.63-71 (1990).

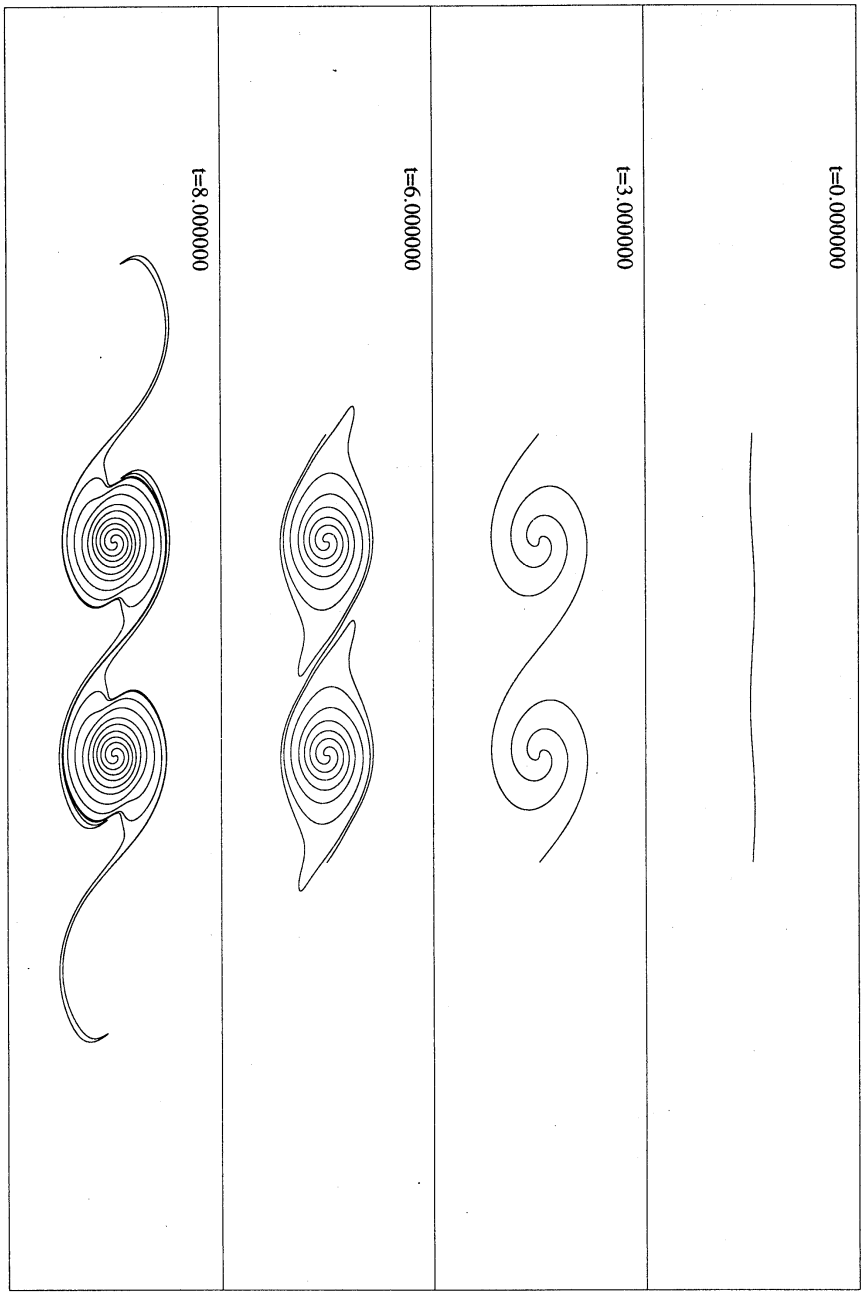


図 10: 渦層の時間発展。時刻 0.0 から 8.0 まで

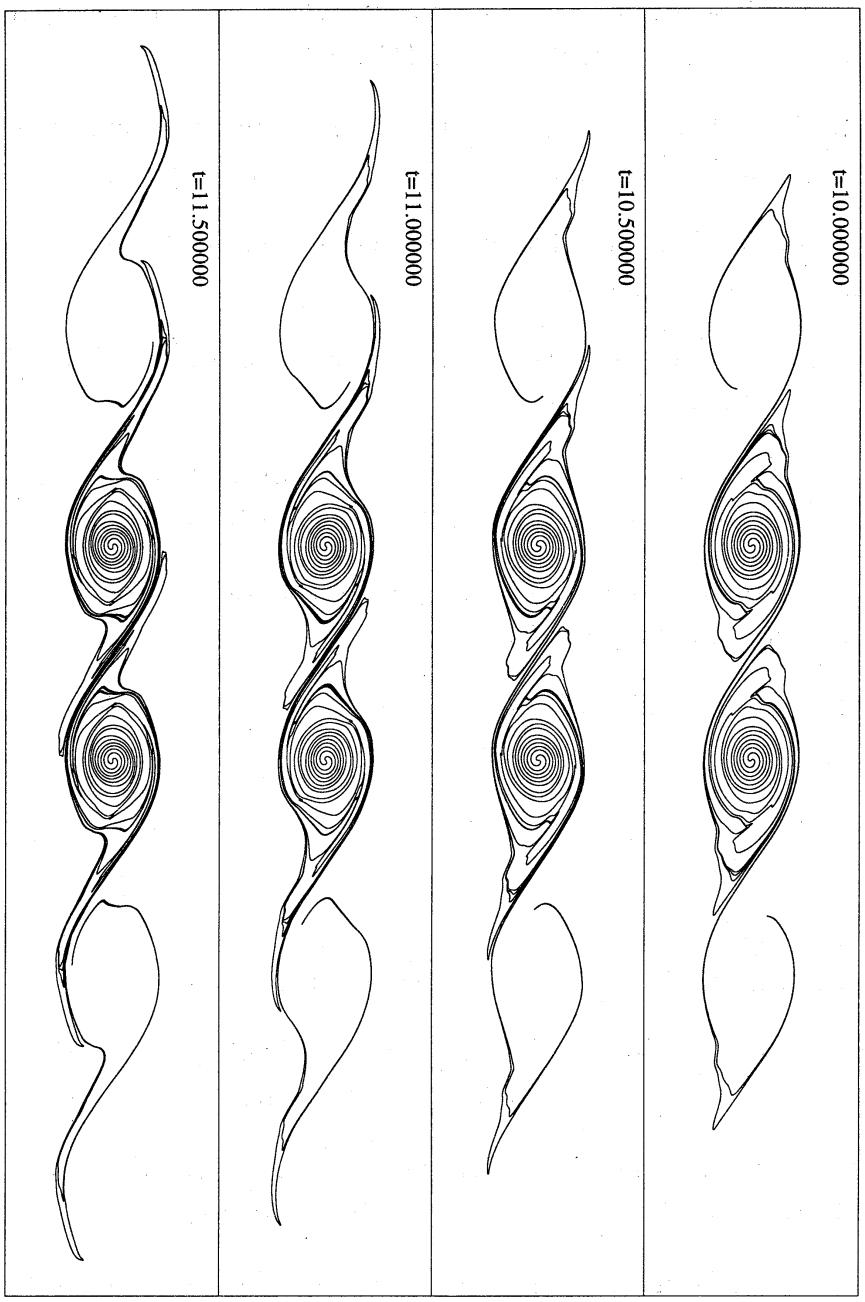


図 11: 渦層の時間発展。時刻 10.0 から 11.5 まで

[8] C.Anderson, An implementation of the fast multipole method without multipole, SIAM J. Sci. Stat. Comput., vol.13 no.4, pp.923-947 (1992).

[9] D.Elliot and J.Board, Fast Fourier transform accelerated fast multipole algorithm, SIAM J. Sci. Comput., vol.17 ,pp.398-415 (1996).

[10] L.Greengard and V.Rokhlin, A new version of the fast multipole method for the Laplace equation in three dimensions, ActaNumerica, vol.6, pp.229-269 (1997).

[11] L.Greengard and J.Strain, The fast Gauss transform, SIAM J. Sci. Stat. Comput., vol.12 no.1, pp.79-94 (1991).

[12] J.Strain, The fast Gauss transform with variable scales, SIAM J. Sci. Stat. Comput., vol.12 no.5, pp.1131-1139 (1991).

[13] L.Greengard and Xiaobai Sun, A new version of the fast Gauss transform, Documenta Mathematica, Extra volume ICM 1998 III, pp.575-584 (1998).

[14] V.Rokhlin, Rapid solution of integral equations of classical potential theory, J.Comp.Phys., vol.60, pp.187-207 (1983).

[15] V.Rokhlin, Rapid solution of integral equations of scattering theory in two dimensions, J.Comp. Phys., vol.86, pp.414-439 (1990).

[16] T.Sakajo and H.Okamoto, An application of Draghicescu's fast summation method to vortex sheet motion, J. Phys. Soc. Japan, vol. 67, No.2, pp.462–470 (1998).

[17] R.E.Caflisch and O.Orellana, Singularity formation and ill-posedness for vortex sheets, SIAM J. Math. Anal. (1986), vol.20, pp.293–307.

[18] R.E.Caflisch and J.S.Lowengrub, Convergence of the vortex method for vortex sheets, SIAM J. Numer. Anal.,(1989), vol. 26, pp.1060–1080.

[19] D.Moore, The spontaneous appearance of a singularity in the shape of an evolving vortex sheet, Proc. R. Soc. A, (1979), vol.365, pp.105–119.

[20] J.Liu and Z.Xin, Convergence of the vortex methods for weak solutions to the 2-D Euler equations with vortex sheet data., Comm. Pure Appl. Math., (1995), vol.48, pp.61–85.